



May 18th, 9:20 AM - 9:55 AM

Positive Identification of LSB Image Steganography Using Cover Image Comparisons

Michael Pelosi

East Central University, Ada Oklahoma, mpelosi@tamut.edu

Nimesh Poudel

East Central University, Ada Oklahoma, nimpou@email.ecok.edu

Pratap Lamichhane

East Central University, Ada Oklahoma, pralam1@email.ecok.edu

Devon Lam

East Central University, Ada Oklahoma, devtlam@email.ecok.edu

Follow this and additional works at: <https://commons.erau.edu/adfsl>



Gary Kessler

Part of the [Aviation Safety and Security Commons](#), [Computer Law Commons](#), [Defense and Security Commons](#), [Embry-Riddle Aeronautical University, Daytona Beach Florida](#), [kessleg1@erau.edu](#)

[Studies Commons](#), [Forensic Science and Technology Commons](#), [Information Security Commons](#),

[National Security Law Commons](#), [OS and Networks Commons](#), [Other Computer Sciences Commons](#), and

the [Social Control, Law, Crime, and Deviance Commons](#)

Scholarly Commons Citation

Pelosi, Michael; Poudel, Nimesh; Lamichhane, Pratap; Lam, Devon; Kessler, Gary; and MacMonagle, Joshua, "Positive Identification of LSB Image Steganography Using Cover Image Comparisons" (2018).

Annual ADFSL Conference on Digital Forensics, Security and Law. 9.

<https://commons.erau.edu/adfsl/2018/presentations/9>

This Peer Reviewed Paper is brought to you for free and open access by the Conferences at Scholarly Commons. It has been accepted for inclusion in Annual ADFSL Conference on Digital Forensics, Security and Law by an authorized administrator of Scholarly Commons. For more information, please contact commons@erau.edu.

EMBRY-RIDDLE
Aeronautical University™
SCHOLARLY COMMONS

(c)ADFSL



Presenter Information

Michael Pelosi, Nimesh Poudel, Pratap Lamichhane, Devon Lam, Gary Kessler, and Joshua MacMonagle

POSITIVE IDENTIFICATION OF LSB IMAGE STEGANOGRAPHY USING COVER IMAGE COMPARISONS

Michael J. Pelosi, Nimesh Poudel, Pratap Lamichhane, Devon Lam
East Central University, Ada, OK, USA

Gary Kessler
Embry-Riddle Aeronautical University

Joshua MacMonagle
Campbell University

ABSTRACT

In this paper we introduce a new software concept specifically designed to allow the digital forensics professional to clearly identify and attribute instances of LSB image steganography by using the original cover image in side-by-side comparison with a suspected steganographic payload image. The “*CounterSteg*” software allows detailed analysis and comparison of both the original cover image and any modified image, using sophisticated bit- and color-channel visual depiction graphics. In certain cases, the steganographic software used for message transmission can be identified by the forensic analysis of LSB and other changes in the payload image. The paper demonstrates usage and typical forensic analysis with eight commonly available steganographic programs. Future work will attempt to automate the typical types of analysis and detection. This is important, as currently there is a steep rise in the use of image LSB steganographic techniques to hide the payload code used by malware and viruses, and for the purposes of data exfiltration. This results because of the fact that the hidden code and/or data can more easily bypass virus and malware signature detection in such a manner as being surreptitiously hidden in an otherwise innocuous image file.

Keywords: Steganography, steganalysis, digital forensics, malware, virus, LSB encoding.

1. INTRODUCTION

Steganography use in malware and for covert communications is on the rise according to many computer security research organizations. Payload images can also bypass data exfiltration mechanisms with relative ease, as positive identification of message-carrying images is elusive.

2. STEGANOGRAPHY SCENARIOS

Steganography has been successfully used for data exfiltration, espionage, concealed communications, command and control for botnets orchestration, malvertising, and ransomware propagation, among other covert and malicious applications. Below is a list of

examples of how steganography can typically operate in various application case scenarios (Gutub, Ankeer, Abu-Ghalioun, Shaheen, Alvi 2008):

- An employee decides to steal sensitive proprietary files. With today's security systems, this would be noticed using classic approaches; however, using steganography, the sensitive files are encoded into images. By doing so, the images can be uploaded to social networks or cloud storage services without triggering red flags.
- A group of cybercriminals is attempting to communicate and synchronize attacks from different countries. Since they cannot go through standard communication channels, they decide to conceal secret messages into profile pictures of social accounts. In this way, they can communicate by uploading and downloading unsuspecting profile photos using whitelisted services.
- A massive botnet has been deployed and is awaiting instructions. Any attempt of communication from a central server to the bots is likely to be discovered eventually. Instead of using a server, the bots are configured to periodically download a feed of text and images from a public social network account. By decoding steganographic data from the feed, instructions are extracted and executed.
- A malicious campaign is planned to affect millions of users, but the perpetrators want to keep it as secret as possible. Since the goal is to exploit a browser vulnerability, they use steganography to conceal malicious code into advertisement images. To reach a large audience quickly, they submit the banner to networks that distribute the image over hundreds of websites. By

doing so the propagation is guaranteed and the campaign revenue can be larger.

- A new ransomware attack hides the communication between the victims and the perpetrator. Using steganography information harvested from the target system is encoded into pictures uploaded to an image hosting website. Using this tactic, the ransomware campaign deployment can remain hidden for a longer period of time.

Tragically, all the examples stated above are based on actual malware case histories. Although many of these attacks were eventually identified and stopped, the amount of time and effort required to detect and stop steganography related attacks and communications was large and continues to be take enormous investigative resources and knowledge. The result is that steganography continues to be a very lucrative technique and opportunity for cybercriminals.

2.1 Rise in the Usage of Steganography for Malware

Recently we have seen steganography used in the following malware programs and cyberespionage tools:

- Microcin, alias Six Little Monkeys;
- NetTraveler;
- Zberp;
- Enfal, which possesses a new loader called Zero.T;
- Shamoon;
- KinS;
- ZeusVM;
- Triton, alias Fibbit.

Why are malware authors increasingly using steganography in their creations? There are three main reasons for this: 1. It helps them conceal not just the data itself but the fact that data is being uploaded and downloaded; 2. It helps bypass DPI systems, which is relevant for corporate systems; and 3. Use of steganography

may help bypass security checks by anti-APT products (as the latter cannot process all image files as corporate networks contain too many of them, and common analysis algorithms are expensive computationally) (Priyanka, Sahoo 2016; Chen, Huygens, Desmet, Joosen 2016; Daryabar, Dehghantanha, Broujerdi 2011).

It is confirmed that Steganography is also used by the malware Vawtrak, Zbot, Lurk, and Stegoloader. In early 2015, Vawtrak started using steganography to hide its settings in favicons. The malware downloads a favicon.ico file from a server hosted on TOR using the tor2web service. This favicon.ico image is the one displayed by browsers at the left side of a URL. Generally, each website contains a favicon.ico image, so security products seeing such requests would typically not test them for validity. Next, the malware extracts a least significant bit from each pixel and constructs a URL for downloading its configuration file (Wyke 2015; Pevny, Kopp, Kfoustek, Ker 2016).

One variant of the Zbot malware also uses steganography to hide its configuration data. This variant downloads a JPEG on the victim's system. The configuration data hides inside this image. Later, the malware extracts the configuration data from the image and performs further malicious actions.

The Lurk malware uses steganography to download other malware onto targeted systems. Instead of simply downloading and executing a malicious binary, Lurk first downloads a BMP image. It uses a least-significant-bit (LSB) algorithm to embed encrypted URLs into the image file. It then extracts the embedded URLs from the image file and then downloads additional malware.

The Stegoloader malware installs malware on victims' systems to steal sensitive information. On successful execution, Stegoloader downloads a PNG image from a legitimate website. It uses steganography to

embed its main module's code inside the downloaded PNG. The malware retrieves the hidden data by applying a steganographic extraction algorithm (Bureau, Deitrich 2015).

The recently discovered Stegano (also known as Astrum) exploit kit, has been used in the past months as part of a very ingenious malvertising campaign. Stegano authors have operated by embedding malicious code inside the RGBA transparency value of each pixel of PNG banner ads. As users viewed the ads, JavaScript code would parse the PNG image, extract the malicious code and redirect the user to the exploit kit landing page, where he would be infected with various types of malware.

Besides Stegano, the second exploit kit discovered in 2016 that heavily relies on steganography is named DNSChanger. The group behind DNSChanger created malicious ads that contained code that launched brute-force attacks against the user's home WiFi router. Attackers were taking control over the victim's router, and injecting ads in all his web traffic. Once again, steganography was crucial to hide this malicious code inside the ads' images, which helped the cybercriminal authors hide the exploit kit's activity from security researchers.

One of the major players operating in the exploit kit market has also turned its efforts to using steganography. The exploit kit's name is 'Sundown,' an exploit kit developed a group of German-speaking developers who called themselves the "Yugoslav Business Network" (or YBN).

Until recently, Sundown operators never bothered to mask exploit code delivered to user files. Security researchers looking at traffic logs could easily identify the Sundown exploit package by looking at URLs, which often contained files ending in ".SWF" or ".XAP" extensions, specific to Flash and Silverlight exploits.

After a recent update, Sundown now hides these exploits as mundane “.PNG” files. The file's header says the file is a PNG image, but its content contains the actual exploit. Sundown traffic is now much harder to detect, and researchers have to put more work in unmasking Sundown operations, just as its operators wanted.

This addition of steganography in Sundown operations was spotted recently and appears to have been inspired by previous three malvertising campaigns. The first is the massive AdGholas malvertising campaign, which ran on the Angler and Neutrino exploit kits, the second is the GooNky malvertising campaign, and the third is a malvertising campaign that delivered the CryLocker ransomware via the RIG exploit kit.

In all cases, the cybercriminals behind these malvertising campaigns had used steganography to deliver PNG images to victims, which contained malicious code that scanned their computer, and later delivered downloaded malware.

The most successful of the malware ad campaigns was the AdGholas campaign, which raged on undetected for almost a year (Cabaj, Caviglione, Mazurczyk, Wendel, Woodward, Zander 2018). The success of those campaigns has apparently convinced the Sundown organization to run many steganography experiments of their own.

By disguising malicious content as PNG files, Sundown is now following the new trend that has slowly taken hold of the exploit kit market in the past year. It is estimated that it will continue to use steganography, at least until security firms find a way to quickly identify malicious PNG files and block them.

Data exfiltration, also known as ‘data theft,’ is the unauthorized transfer of sensitive information from a computer or a server. In 2016, there were attacks related to Magento, an

online e-commerce platform. The attacks used image steganography to hide payment card details. After execution, the malicious code collected the payment card details and hid this inside a local image file, such as an actual product picture. Once data collection was completed, the attacker then downloaded the image file (typical for an e-commerce website) and extracted the hidden data (Melanson 2017).

These trends described above in particular suggest that malware writers are on the verge of adopting steganography on a mass scale. Most modern anti-malware solutions provide little, if any, protection from steganography. As a result, any steganographic carrier file such as a digital image or even a video file, that can be used to conceal stolen data, or communications between a malware program and a command and control server, will remain undetected.

2.2 Improved Detection Procedures and Techniques

Many statistical techniques have been developed over the years to attribute a probability of a file being a steganographic cover file, but owing to the various methodologies and data payload densities, these methods can be considered unreliable at best. In certain cases, regarding digital image steganography, it may be possible to visually or algorithmically determine the original image. This would be the image file before payload injection takes place.

Normally, image steganography will involve altering the least significant bits (LSBs) in the cover image. By comparing the original image LSBs to the payload image LSBs, a positive identification cannot take place indicating the use of steganography. In performing steganalysis, possessing the payload image file alone leaves little option than the use of statistically based probabilistic tools for attribution (Walia, Jain, Navdeep 2010).

However, by locating the original digital image file for comparison, the alteration of LSBs alone is quite the forensic "smoking gun," so to speak, for reliable attribution for the use of steganography software to send messages or data files. This could be a starting point for further investigation for law enforcement or other investigative authorities.

In performing good steganographic procedure, a suspect will take care to data wipe any original cover file, to prevent such a comparison from taking place. However, in practice, human error and technical limitations may prevent completely effective data erasure of the original cover image. In that light, we recommend an active search for the original image if suspicion of steganographic usage exists. There are many possibilities for locating the original image that will allow later positive attribution for the use of steganography. The following lists many examples of the large number of potential locations to find an original image for comparison.

Places to find non-payload (also known as a "cover") image:

1. Suspect hard drive filesystems
2. Suspect removable USB drives
3. Suspect cameras and mobile devices
4. Suspect CDs and DVDs
5. Local email inboxes/outboxes
6. Cloud email inboxes/outboxes
7. Recent web search and browser histories
8. Google image searches
9. Network attached storage devices
10. Employment computers and networks
11. Recycle bins
12. Deleted files removed from recycle bins
13. Online photo galleries
14. Personal and business associates' files as listed above

Any image presenting the same visual appearance and pixel dimensions is an excellent candidate to be the original cover image file. In

that light we offer novel digital forensics software to allow the investigator to perform a quick and more convenient LSB analysis in comparison to make a positive identification. Further, the LSB comparison image results can be conveniently copied and pasted for reports documents for conclusive proof of the use of steganography.

3. THE "COUNTERSTEG" SOFTWARE

The software introduced with this paper, *CounterSteg*, is available free of charge from the following website: <http://199.175.52.196/CounterSteg/>. This Windows-based software allows the loading of two images and comparison of pixel color bits in the LSB plane. The software will also run under Linux with Wine installed. Detailed analysis is performed visually at the moment; however, we envision future algorithms which can automate the results conclusion positively. The software can be used to detect differences between original and payload image. Positive detection implies, in general, identical pixel dimensions and most pixels identical except for various LSB values. Human forensic analysis does confirm final analysis using additional informed investigation techniques described in this paper.

3.1 Recommended Usage

If a suspect image is detected, a search should be conducted for visually similar and pixel dimensionally identical images in the locations listed above, among others. Once potential matches are identified, the software should be used to look for differences in the LSB and perhaps nearby planes. If such differences are detected, it can be considered positive identification for the use of steganography, although it is unlikely the original message or data can be recovered, except with the cooperation of the suspect, or acquisition of the

original software and/or encryption key used to embed the data; however, with positive results in hand, this further investigation or warrant acquisition can be embarked upon with great confidence.

3.2 Results Using Various Steganographic Programs for Experimentation and Analysis

The following are the results of steganography analysis and detection with commonly available tools that may be used by a cybercriminal, or other suspect, free of charge. These programs include:

- OpenPuff -
http://embeddedsd.net/OpenPuff_Steganography_Home.html
- Steganography Online -
<http://stylesuxx.github.io/steganography/>
- Geocaching Toolbox -
<https://www.geocachingtoolbox.com/index.php?page=steganography>
- OTP-Steg -
<http://www.mauisolarsoftware.com/OTP-Steg/>
- f5stego.js -
<http://desudesutalk.github.io/f5stegojs/>
- DevFarmSteganography -
<https://devfarm.it/steganography/>
- StegoShare -
<http://stegoshare.sourceforge.net/>
- BitCrypt - <http://bitcrypt.moshe-szweizer.com/>
- OpenStego -
<https://www.openstego.com/index.html>
-
- Other programs unable to be tested at this time were:
- Steghide -
<http://steghide.sourceforge.net/download.php>

- SteganPEG -
http://download.cnet.com/SteganPEG/3000-2193_4-75914262.html
- ManyTools Steganography -
<https://manytools.org/hacker-tools/steganography-encode-text-into-image/>
- Steganographic Encoder (Steghide) -
<https://futureboy.us/stegano/encinput.html>
- Mobilefish -
<https://www.mobilefish.com/services/steganography/steganography.php>
- Kwebbel -
<http://www.kwebbel.net/stega/enindex.php>

The authors will attempt tests with untested and additional tools soon.

As examples of the type of forensic steganalysis that can be conducted with *CounterSteg*, we have embedded text data into a standard cover image using some of the above listed and easily available steganographic programs. Each of these programs was accessed directly from a website or downloaded executable. Each took less than 10-20 minutes to use to embed the standard text data into the standard cover image, which is shown on the next page in Figure 1.

The standard cover image shown below was taken by one of the authors in Keyser, West Virginia in the Fall of 2015, using a Nikon D90 digital camera. This image shows a fairly even distribution of red, green, and blue colors throughout the image, except for the center top open to the sky. In this specific area, the camera CMOS sensor saturated to white (RGB[255,255,255]), and each of the pixels here represents that single saturated white color. This is notable for LSB steganography in that steganographic programs that modify pixels in this area will be more easily statistically detected. Alterations to the LSB values in pixels

in solid color, or saturated, portions of the image are a good indicator of nonstandard modifications (such as steganography). Good

steganographic programs will attempt to avoid modifications to these specific areas, among others.



Figure 1. Cover image taken with Nikon D90 camera.

The standard text embedded in the image was the President Kennedy inauguration speech, which is 1,366 words, and 7,512 characters. The size of the text is 7.38 KB (7,566 bytes), file size on disk was 8.00 KB (8,192 bytes). Kennedy's inauguration speech was delivered on January 20, 1961.

The *CounterSteg* program produces detailed visual analysis and comparison of digital images, specifically in each color and bit-plane, in addition to combinations of colors in a particular bit-plane. The figure below shows that the image analysis window, which calculates results for 45 various bit plane and color combinations. The Alpha channel is the transparency channel and remains on unused in many images.

Surprisingly, however, some steganographic programs make spurious or data-carrying modifications in the alpha channel, so it is important to also keep an investigative eye on this color channel. In the analysis window shown below, each color channel is broken down by bits, with bit 0 corresponding to the LSB, and bit 7 corresponding to the MSB. The values in each bit plane are shown for red, green, and blue channels, as well as the alpha channel. For the graphic shown for "All Bits," the pixel color here will be non-black if any of the bits (0-7) is set to a nonzero. The color value is the relative intensity of that color (red, green, or blue) in the range of 0 to 255.

Finally, the image shown in the grid in the upper-left for "All Bits" and "All Colors" is basically the original image, since it shows the combined color values in all channels and all

bits. Any of the images shown in the grid can be clicked on to bring up a new window showing that image full-size. This can be copied and pasted into an image editing program for further analysis or the saving of the image.

The overall idea and philosophy of the *CounterSteg* steganalysis software is to allow the convenient analysis and comparison of before and after images to look for the telltale traces of steganographic software activity and modifications. In many cases, these follow similar patterns, and the forensic analysis conducted can make informed conclusions based on typical similar patterns from the various categories of steganographic software currently available. The software available generally falls into several categories, which for the purposes of this paper we will categorize as: 3. the good, 2. the bad, and 1. the ugly.

We will start showing telltale traces from "ugly" steganographic software, typically this

quality of software can be easily detected even without the original cover image for comparison. Even in the case of good steganographic software, having the original cover image on hand makes positive identification of the activity highly probable.

The cover image analysis window shown below clearly depicts the area of white saturation in the upper center of the image (the area open to the sky through the trees). Other color and blue channels show a reasonable distribution of intensities throughout most areas of the image. Ideally, for steganographic activities, areas of solid colors, saturation, and low noise between colors and shades should be avoided. This is to circumvent statistical analysis of the steganographic payload carrying image that may indicate a high probability of data carrying modifications.

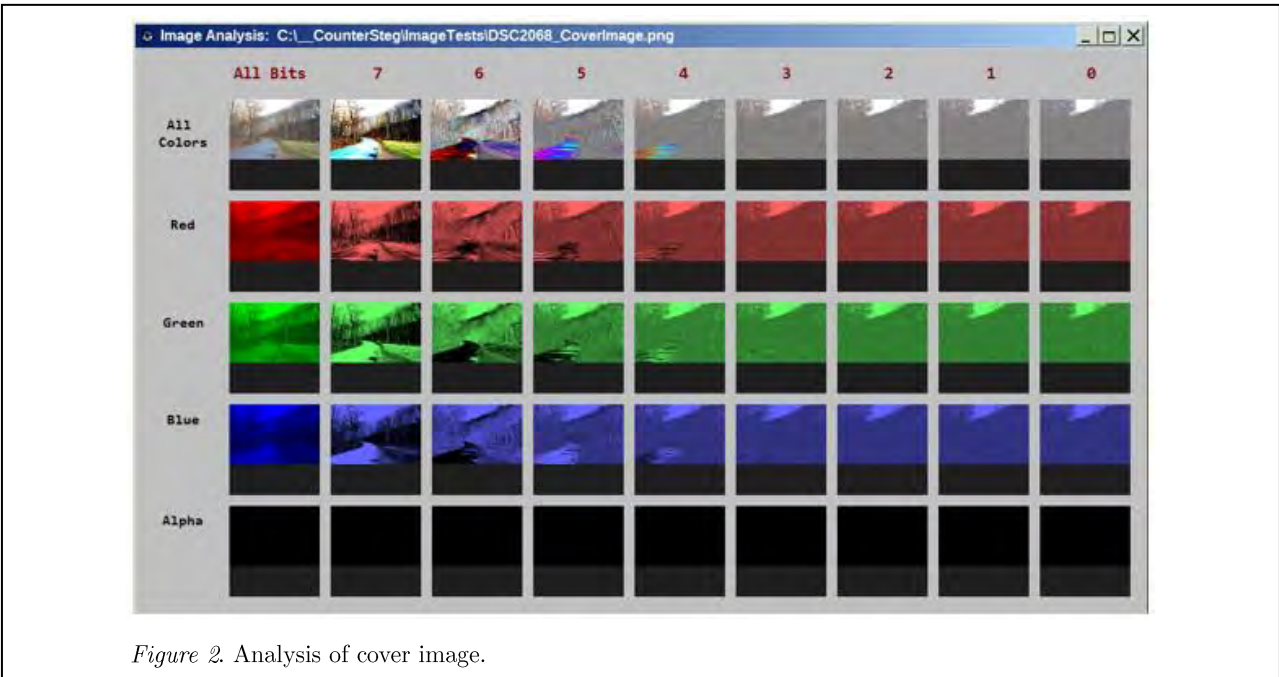


Figure 2. Analysis of cover image.

Clicking on the LSB (bit 0) image for all colors, the image below is brought up and is shown as Figure 3. This image shows the bits colored for whether red, green, or blue pixel

LSBs are sent to 0 or 1. Various color shades are shown depending on multiple values, however if only one bit is set, such as red, the pixel will remain red as shown in the red bit 0 color image

shown as part of Figure 1. Using the LSB image can give an idea of the relative distribution of LSB values in the image in various colors. In general, for many photographs, the distribution will be largely random except for saturated or

solid color areas of the image. This image is the starting point for our further analysis, as modifications to the LSB bit plane are particularly evident in certain qualities of steganographic software available.

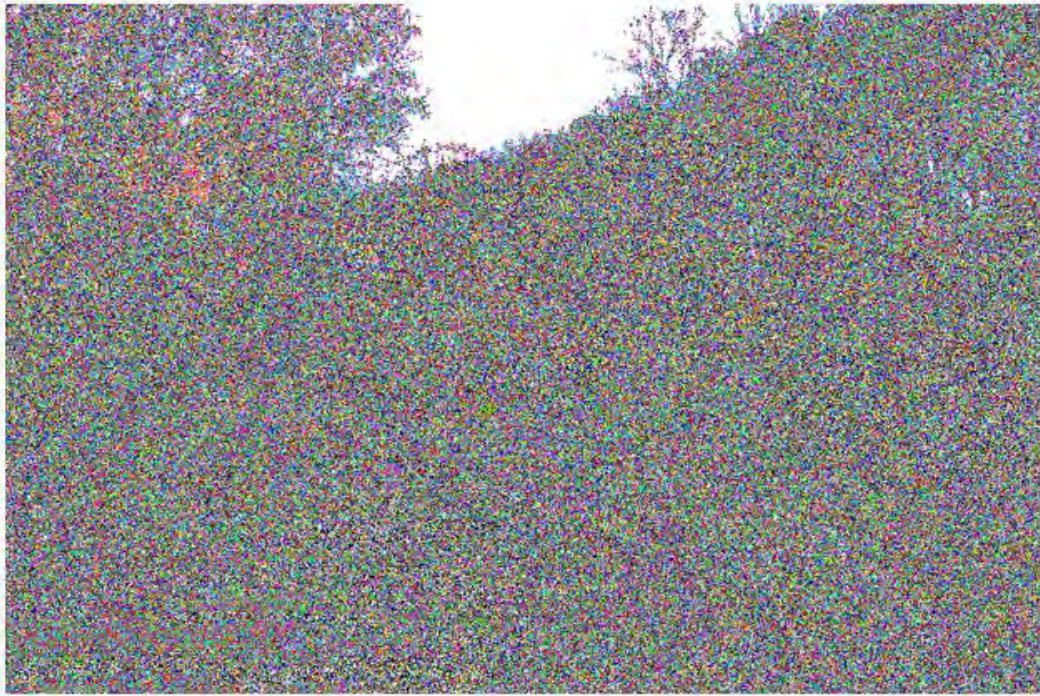


Figure 3. LSB values analysis of cover image.

Also contained within the *CounterSteg* is the ability to visually show pixel variations between colors in a local area. Below, in Figure 4 is an image that shows the variation in the green color channel. Areas of lighter colors indicate higher variation (which could be considered noise) between pixel colors in nearby areas of the image. Since the area of the sky is black — this indicates no variation and LSB, or other bit

planes — modifications will be more easily statistically detectable here.

The figure below depicts the green color channel variation using all bits. Variation is based on the calculation of peak signal to noise ratio (PSNR) of each pixel green color versus its neighbors.



Figure 4. Pixel green color variation throughout the cover image.

In addition to *CounterSteg* providing bit by bit and color analysis for a single image, the software also allows image comparisons using a similar breakdown. The comparison of the original cover image to a contrast adjusted image is shown below. In this breakdown, only differences between pixels, colors, and bits are shown. Below each comparison image is shown the total number of bit, color, and/or pixel differences (depending on the analysis), as well as the percentage of changes relative to the total number of changes possible.

This analysis window allows quick and convenient comparison between an assumed original cover image, and the assumed steganographic payload image. The specific type, location, and scale of the differences can help to clearly identify steganographic activities

that have been performed on the image, the likely image payload size, and perhaps even the likely specific steganographic software that has been used in certain cases. In the following narrative, we will detail forensic profiles of various software packages and their results.

3.3 Contrast Adjustment

As an example of a standard image adjustment in comparison, below is the results of comparing the original cover image to a modified image with a small contrast adjustment. In addition to large modifications in the LSB plane we are also seeing large to small modifications in all other bit planes, and in all colors. In general, if two visually similar versions of an image exist — seeing changes like this in comparison would generally not indicate steganographic activities.

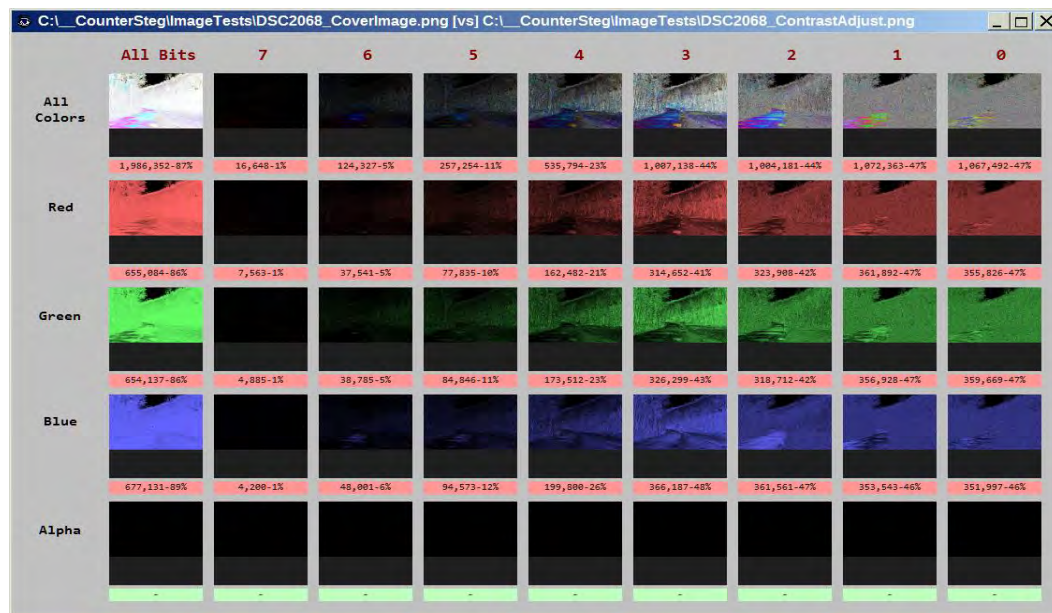


Figure 5. Comparison of cover image to contrast adjusted image.

3.4 Brightness Adjustment

Similar to the small contrast adjustment, a brightness adjustment of the original image

results in a comparison profile showing large modifications in all colors and in all bit planes.

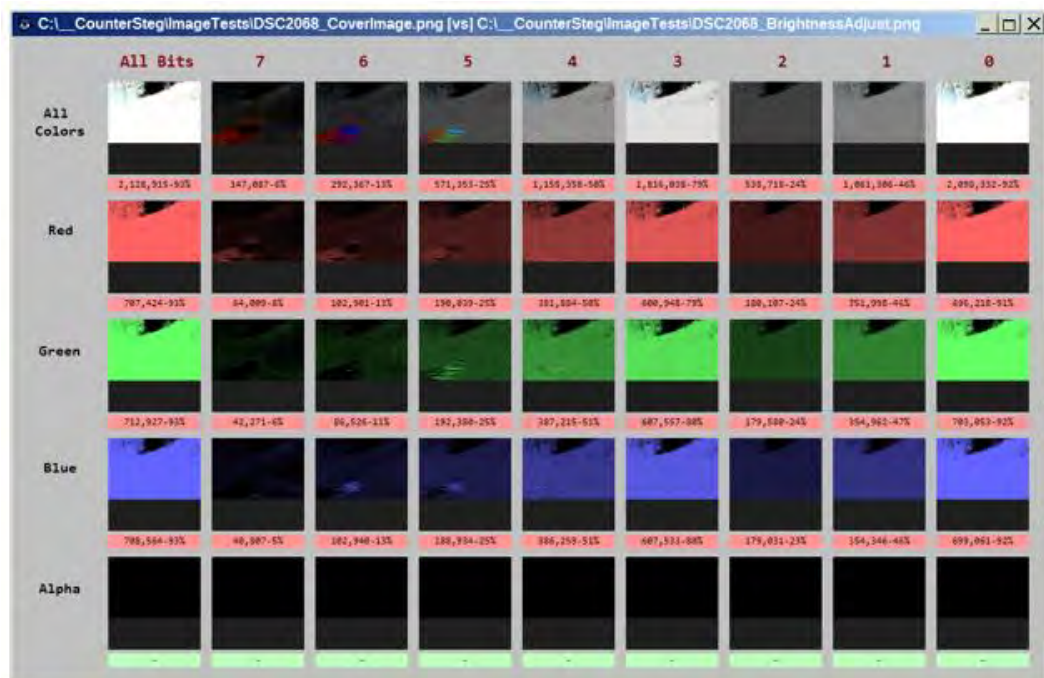


Figure 6. Comparison of cover image to brightness adjusted image.

3.5 Gamma Adjustment

The Gamma adjustment color filter compresses or stretches various colors and would result in a comparison profile similar to the one shown below. All colors and all bit planes are greatly modified. The differences between the two images will be visually apparent.

In conclusion, standard image filters, such as those found in Photoshop, or other image

editing software, for contrast, brightness, and gamma adjustment, do not generally yield comparison results similar to what we will depict in the following narrative for steganographic related changes. This is with one exception, BitCrypt, which should still be detectable using other digital forensic clues and analysis.



Figure 7. Comparison of cover image to gamma adjusted image.

For the purposes of this paper and analysis, we will divide steganographic software into three general categories ranging from the ugly to the good and including the bad in between. Each software embeds a digital payload with varying levels of detectable qualities. The ugly is the worst, and should result in the easiest forensic detections, even without possessing the original cover image. In the case of good software, embedded data will be virtually undetectable without the original cover image for comparison.

However, in the case of good software, with the cover image original on hand, a forensic bit plane comparison makes the activity even then *easily evident*. This is in comparison to typical image changes shown from standard image filters, such as contrast, brightness, and gamma adjustment, explained previously. Possession of the original cover image makes positive identification and attribution possible in almost all circumstances.

4. THE UGLY

Typically, ugly software takes a "let it rip" attitude and just shoves the data to be embedded into the LSB image color plane, without regard to how easily this would be possibly detected using a forensic analysis.

4.1 Steganography Online

<http://stylesuxx.github.io/steganography/>

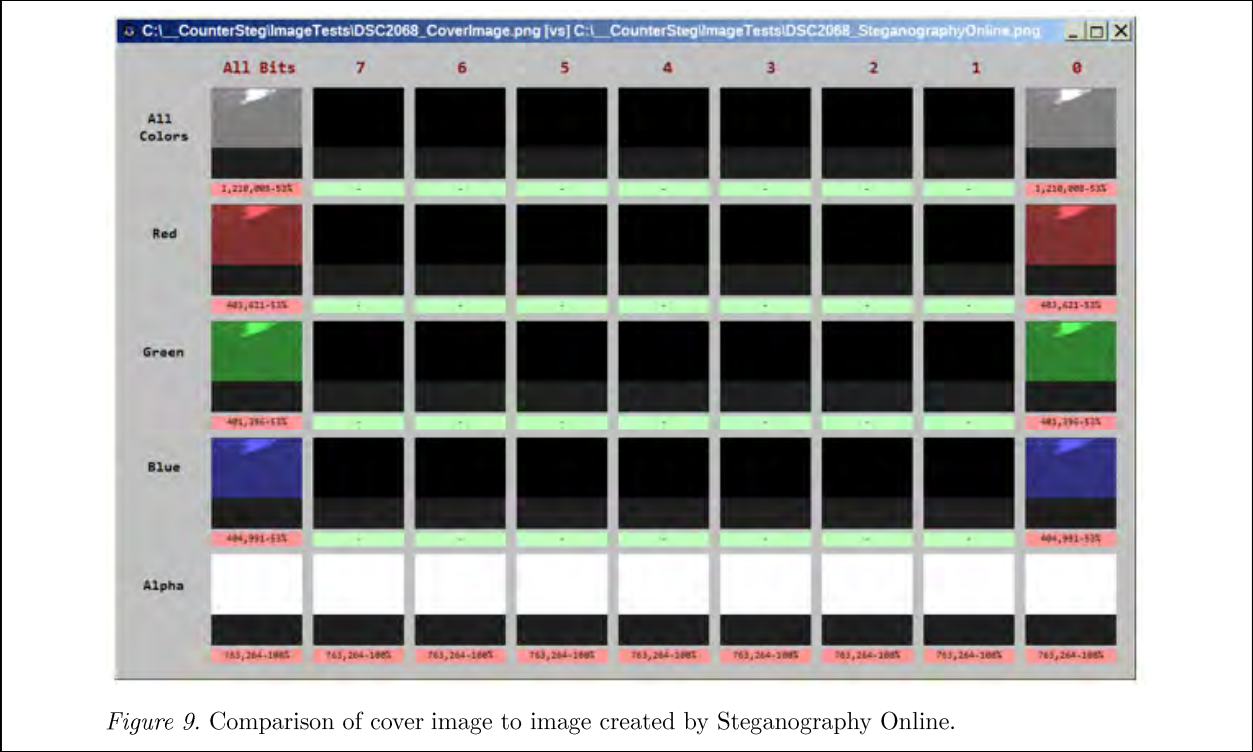
This particular software apparently first completely zeroes out the LSB bit plane in all colors, and then encodes the data into a narrow strip at the top of the image. This is visually evident in the analysis image shown below — where all LSB bits are simply set to zero, except for the data at the top. This is the least sophisticated and most naïve of all the steganographic software we will be analyzing — the ugly.



Figure 8 Analysis of image created by Steganography Online

In the comparison image shown below, you can see all bit planes are identical from bit plane 7 to 1. All modifications take place in bit plane

0 (LSB), which is in fact first set to zero, and then the data is encoded.



The image shown below is an expansion of all colors in bit plane 0, the data containing strip at the top is easily evident. In particular, the area of saturated white pixels at the top is

completely overwritten. This software will be easily forensically detectable and identifiable in usage.



4.2 StegoShare

<http://stegoshare.sourceforge.net/>

The next ugliest software uses a similar approach; however, does not completely zero out the LSB plane. In addition, it makes modifications to bit plane 1 and 2, for reasons

unknown. Other bit planes remain unchanged. Further, some type of narrow strip of information is embedded in the top center of the image. Analyzing the image in bit planes 0, 1, and 2, as shown below, clearly depicts the modifications.

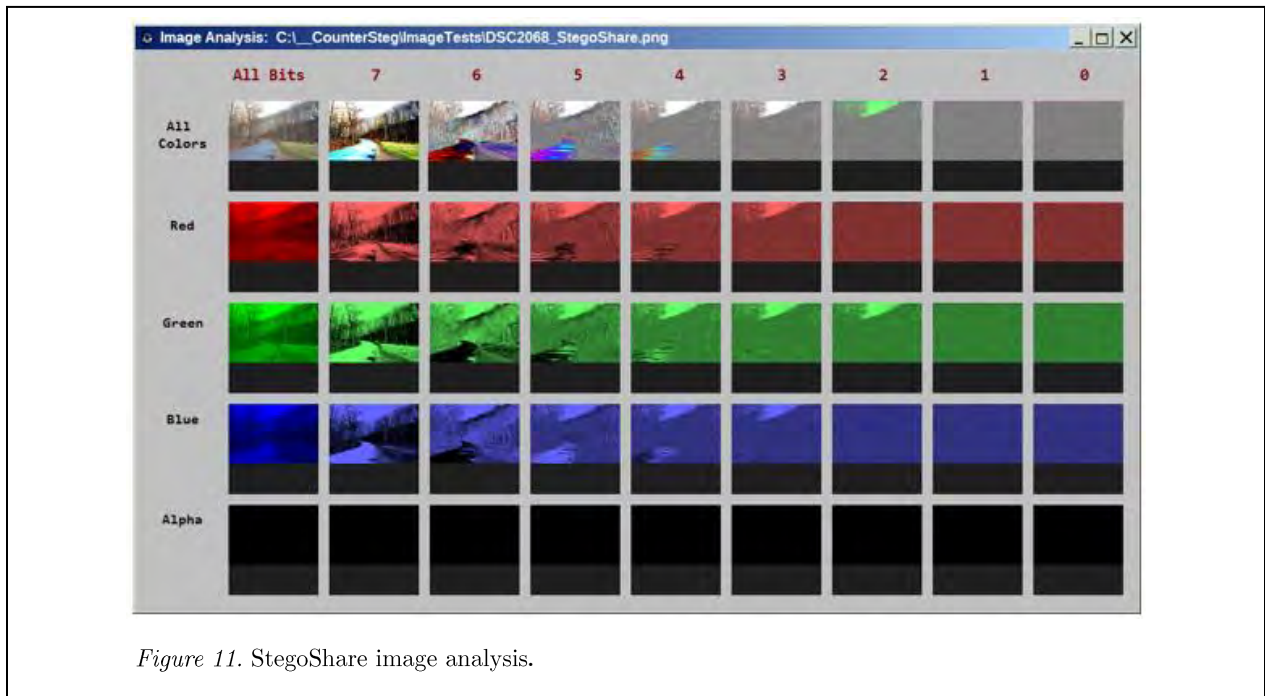


Figure 11. StegoShare image analysis.

In comparing the images, you can see the large percentage (33-50%) of modifications

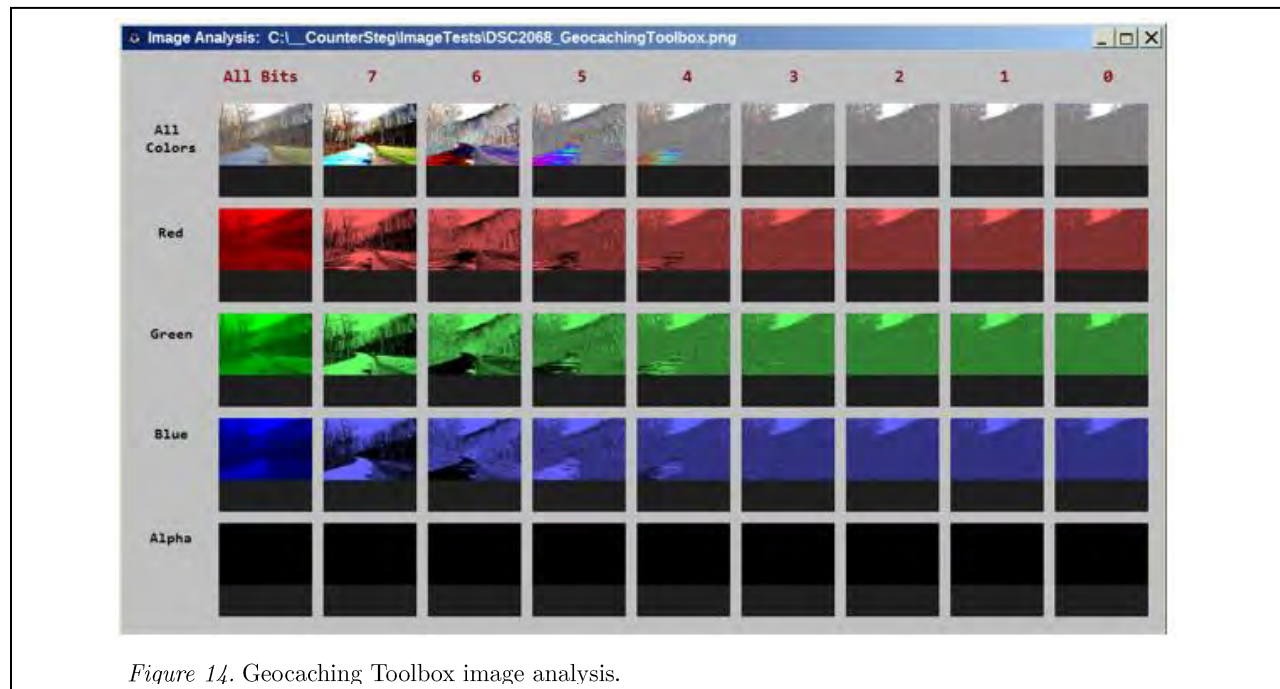
made to bit plane 0, 1, and 2, with the exception of green in bit plane 2.

4.3 Geocaching Toolbox

<https://www.geocachingtoolbox.com/index.php?page=steganography>

This software only alters the LSB bit plane in all three colors; however, it includes all the data into a narrow strip at the bottom of the image. This would be easily detectable using RS

statistical analysis as changes to the bit patterns in only a small fraction of the image (the portion containing the data). The overall analysis of the image, as shown below in Figure 14, does not indicate much forensically. In this case, we also need the original cover image for comparison, which then makes the positive conclusion evident.



Below in Figure 15 is the comparison image, showing the data payload embedded to the narrow strip in the bottom of the image. However, due to the small amount of pixel changes (1%), it is likely at least the software compresses the data before embedding.

Compressing the text data before embedding typically can reduce the size of the necessary modifications to the image by 80 to 90%. As a result, higher quality steganographic software will always compress data before engaging in the image embedding process.

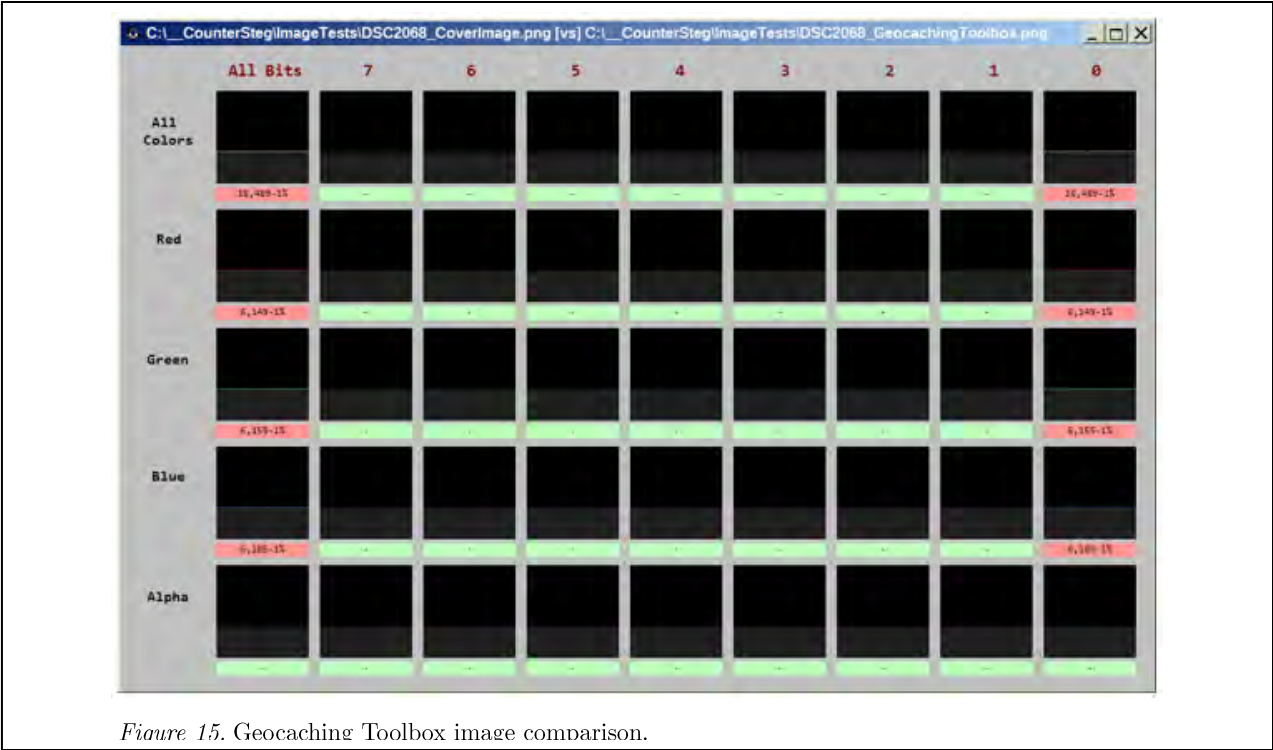


Figure 15. Geocaching Toolbox image comparison.

The narrow strip is clearly visible in Figure 16 at the bottom.



Figure 16. Geocaching Toolbox LSB changes.

4.4 DevFarmSteganography

<https://devfarm.it/steganography/>

This software is comparable to the previous software, Geocaching Toolbox, and may make

use of the same steganographic embedding libraries. The modifications to the image are very similar, with only slight differences in the number of pixels modified.

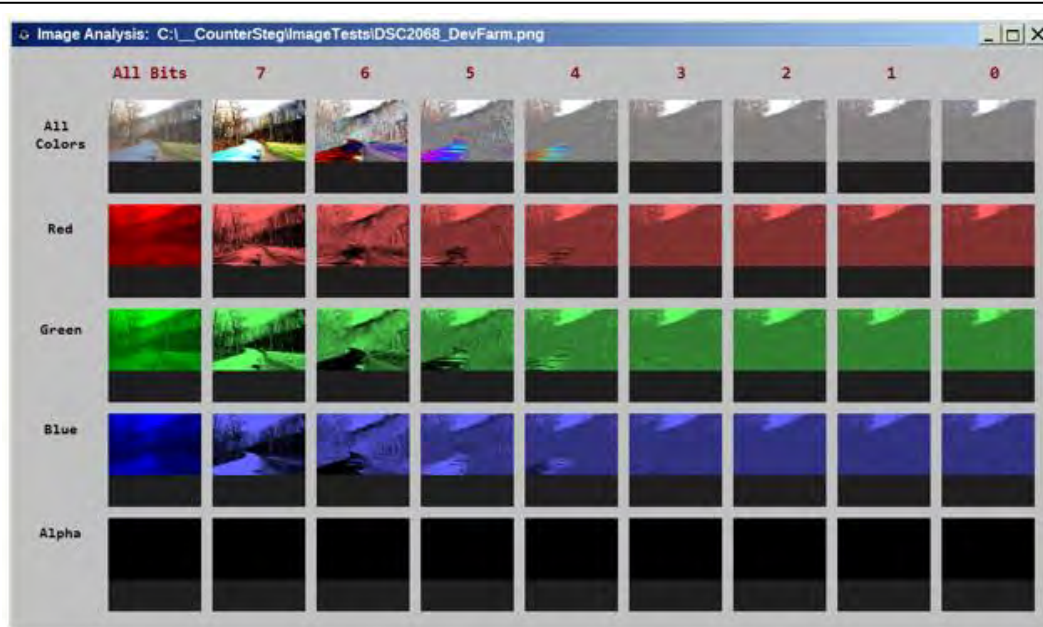


Figure 17. DevFarm Steganography image analysis.

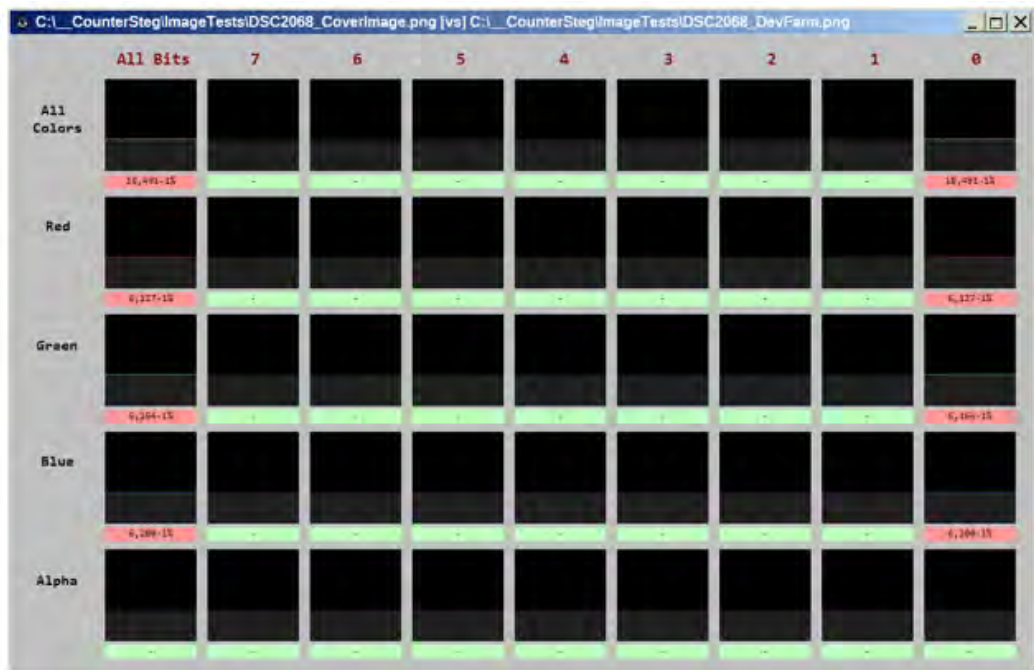


Figure 18. DevFarm Steganography image comparison.



Figure 19. DevFarm Steganographyv LSB image changes.

5. THE BAD

The best software is an improvement over the ugly, at least narrow strips of pixels are not apparently encoded (and more easily detected), however security shortcomings are still evident.

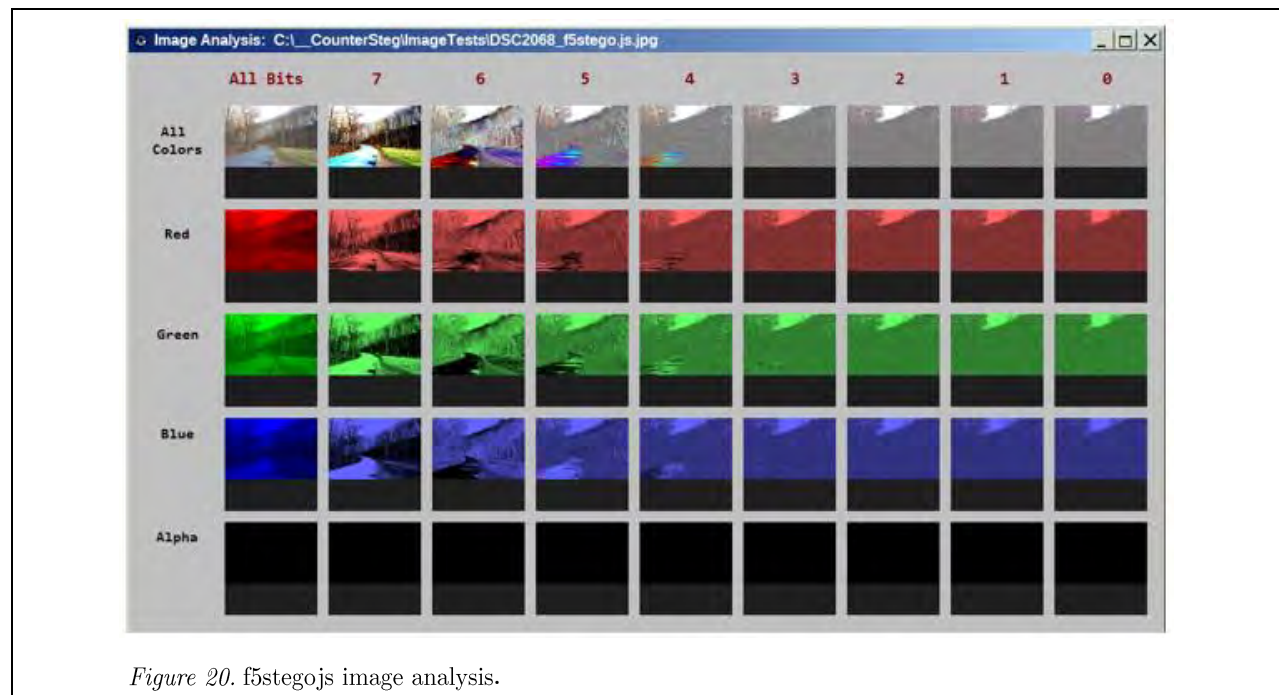
5.1 `f5stego.js`

<http://desudesutalk.github.io/f5stegojs/>

This software seems to take the unique approach of ignoring the LSB and simply

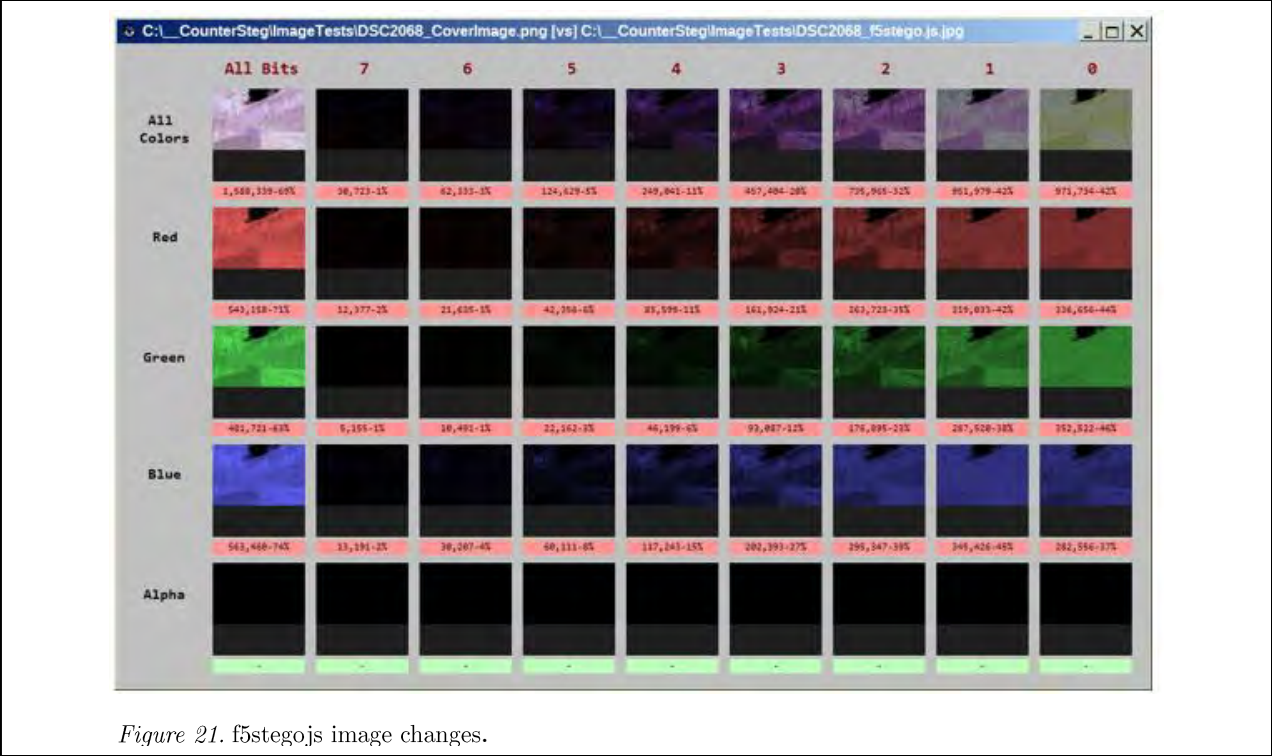
encoding data depending whether color values are odd or even. Even though many of the pixel colors are modified through multiple bit planes, some of the color values remain unchanged. This is evident in the comparison image of all bits in all colors shown as Figure 21.

An analysis of the image shown in Figure 20 does not show undue pixel modifications or strips; however, due to the large amount of color changes, visual differences will be evident between the original and modified image.



Below, in Figure 21, is the comparison analysis. Notice that all bit planes and all colors are modified. On first analysis, this would appear to be very similar to the contrast adjustment shown previously, however in that case virtually all pixels are modified, except for the saturated white area. In this case, many pixels remain unchanged, hinting at the possibility of "all bit" encoding. In other words, data is encoded by overall color intensity value

for the respective color channel, red, blue, or green (in the range 0 to 255). Depending on whether the color intensity is odd or even, this indicates the value of the bit for that particular color channel. Three bits can be encoded for each pixel in this fashion; however, forensic analysis of the steganographic image easily identifies a payload because of the fact that only selective pixels are modified.



As shown below in Figure 22 in the expansion of the comparison image in all color LSB bits — many pixels remain unchanged in an apparent random fashion. It is likely saturated pixels are avoided (an aspect of

quality in this software), and random noise is added in areas of the image not needed for further data encoding. Overall, 69% of the pixel color values in this image have been modified.

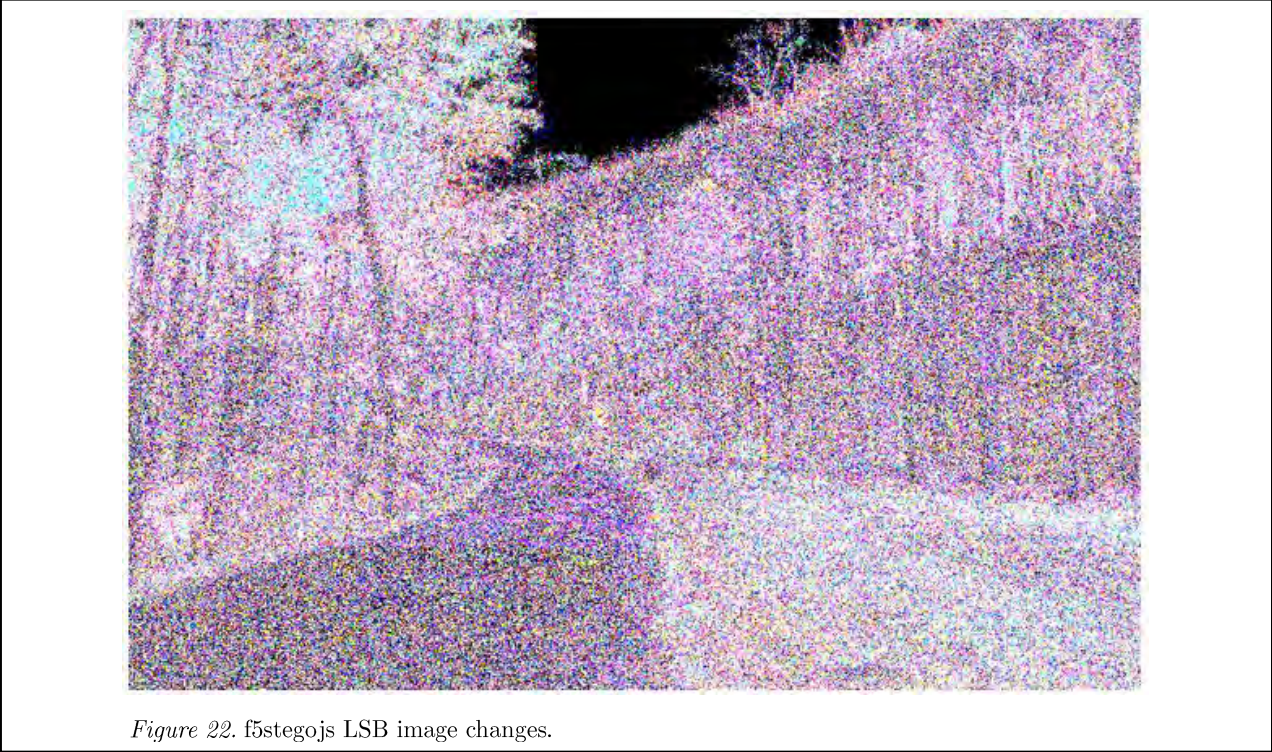


Figure 22. f5stegojs LSB image changes.

5.2 BitCrypt

<http://bitcrypt.moshe-szweizer.com/>

BitCrypt fails in the area of modifying saturated white pixels and makes this software more easily detectable. The sophisticated forensic analyst will be aware that camera CMOS sensors typically saturate to maximum values in bright areas, such as the sky, will not vary between colors pixel to pixel. These block

areas will be fully saturated and will remain so through the area of the similar object, such as the overcast sky. Modifications to pixels in these areas will be telltale signs for software image message or data carrying modifications.

An overall analysis of the image shows the bit 0 and 1 modifications in the area of saturation. Bits 2-7 appear to be largely unchanged in the area of saturation.

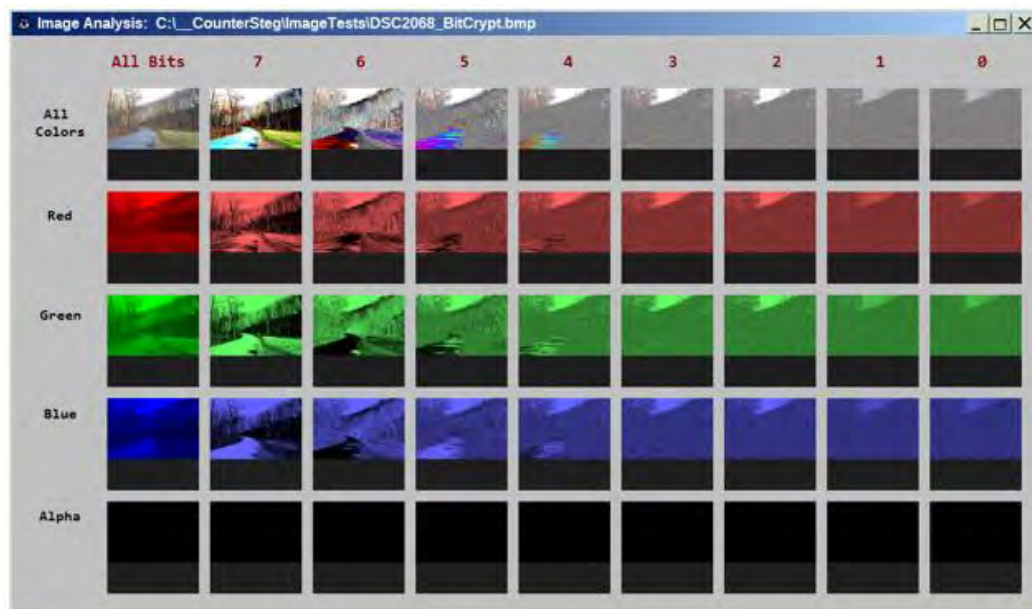
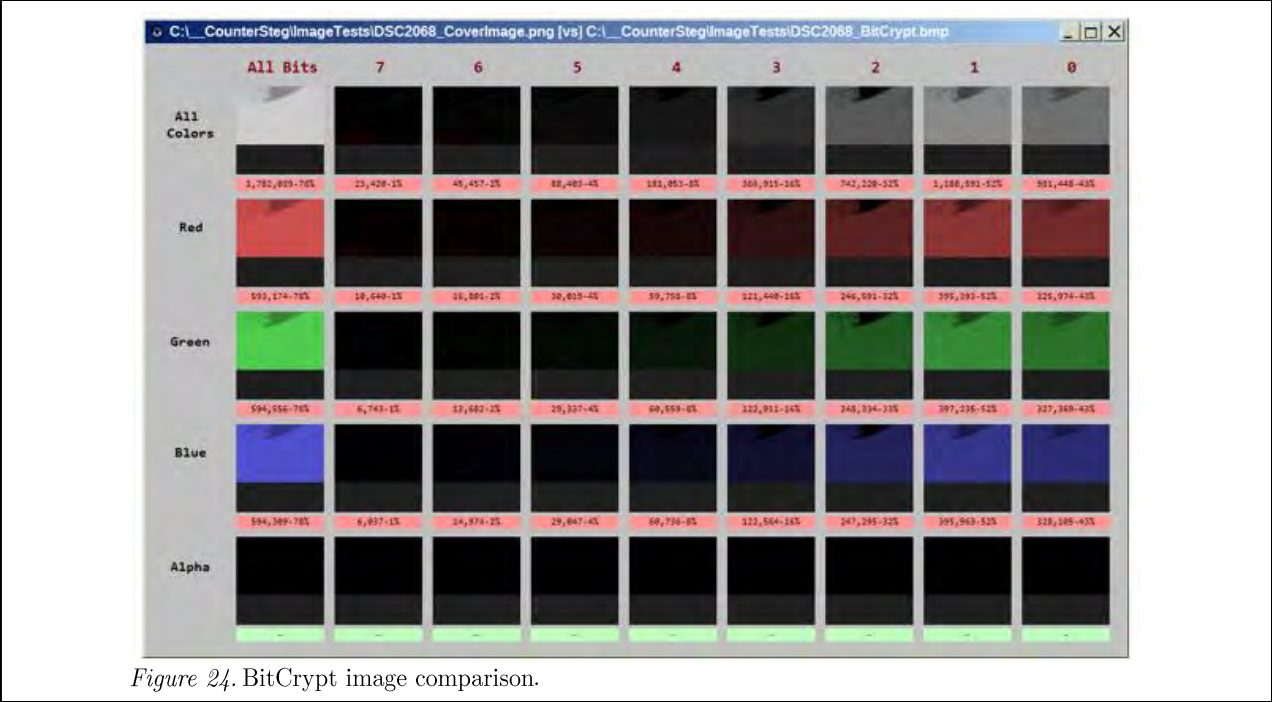


Figure 23. BitCrypt image analysis.

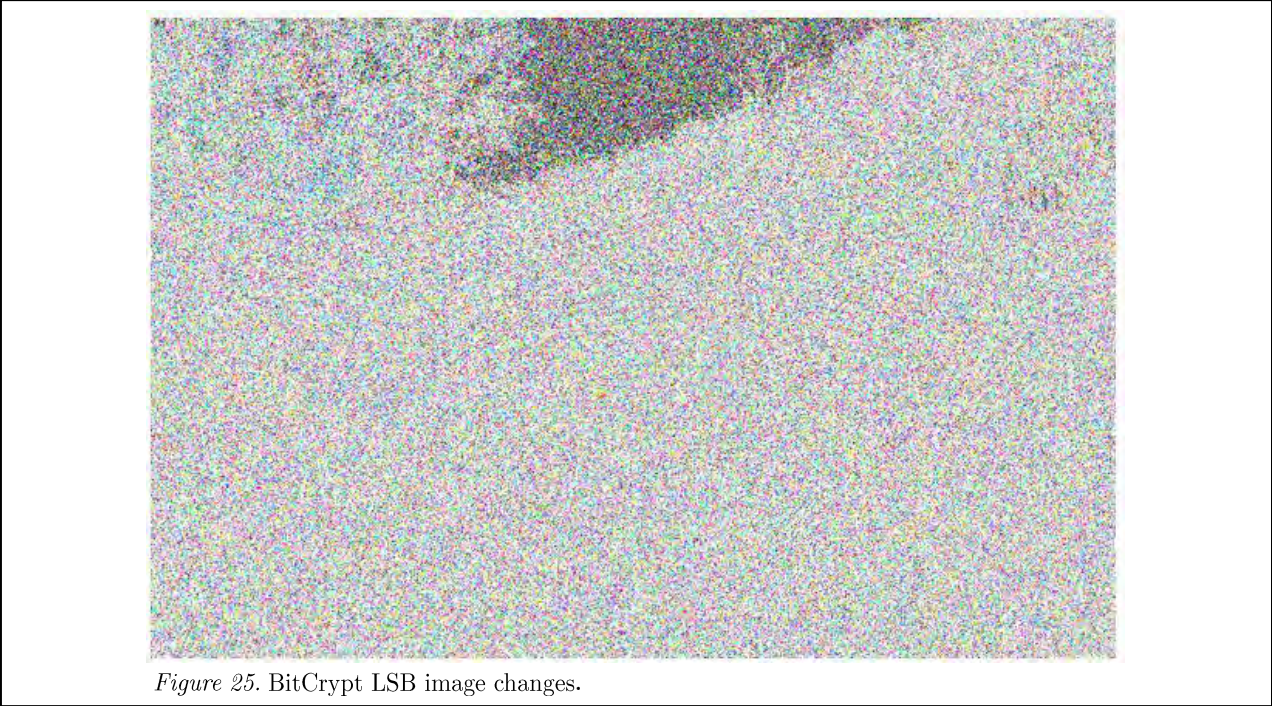
Comparison with the cover image shows the broad modification of pixel colors in bits 0 through 7; however, the area of saturation is avoided above bit 1. This most likely is a software coding implementation to create a

similarity with a standard image processing function, such as brightness, contrast, or gamma adjustment; however, in those cases, the changes in the saturated area will propagate up through bit 7, and including bit 7.



Further, the expansion of the bit 0 in all colors graphic analysis depicts the seemingly random dispersal of LSB bit changes, except for the relative lack of changes in the saturated area. This particular forensic pattern for steganographic activity is indeed unique. While subtly different from standard image processing

comparison results, the bit modification pattern should still be able to be classified and identifiable. Further test images should be created, and comparisons conducted in the future to analyze and identify with further clarity the modifications BitCrypt makes to images.



5.3 OpenPuff

http://embeddedsd.net/OpenPuff_Steganography_Home.html

While completely avoiding the saturated pixel area, which is good, OpenPuff attempts to modify far too many pixels to not be detectable especially with a comparison image in hand. The software only modifies pixels in the LSB plane, making it virtually undetectable visually. However, overall 12% of the pixels are modified

in the image — compare this to 1% even with less sophisticated programs. Statistical analysis of the image will reveal changes to typical photograph LSB bit patterns in typical similar photos with similar CMOS sensors.

Overall analysis of the image, as shown below in Figure 26, does not reveal any particular fine points for analysis — making OpenPuff the best of the bad software for Steganography in this paper analysis.

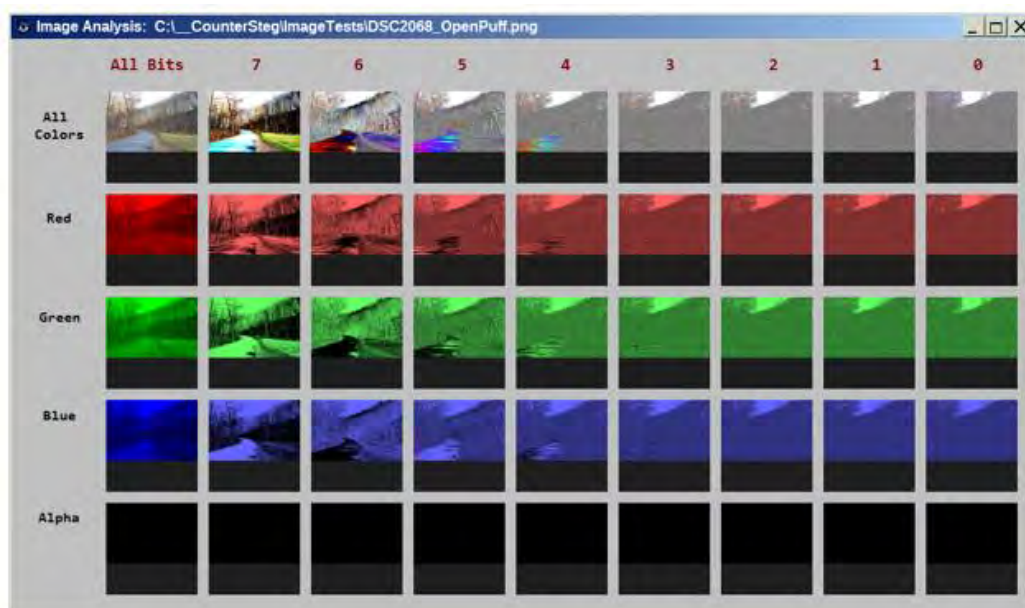
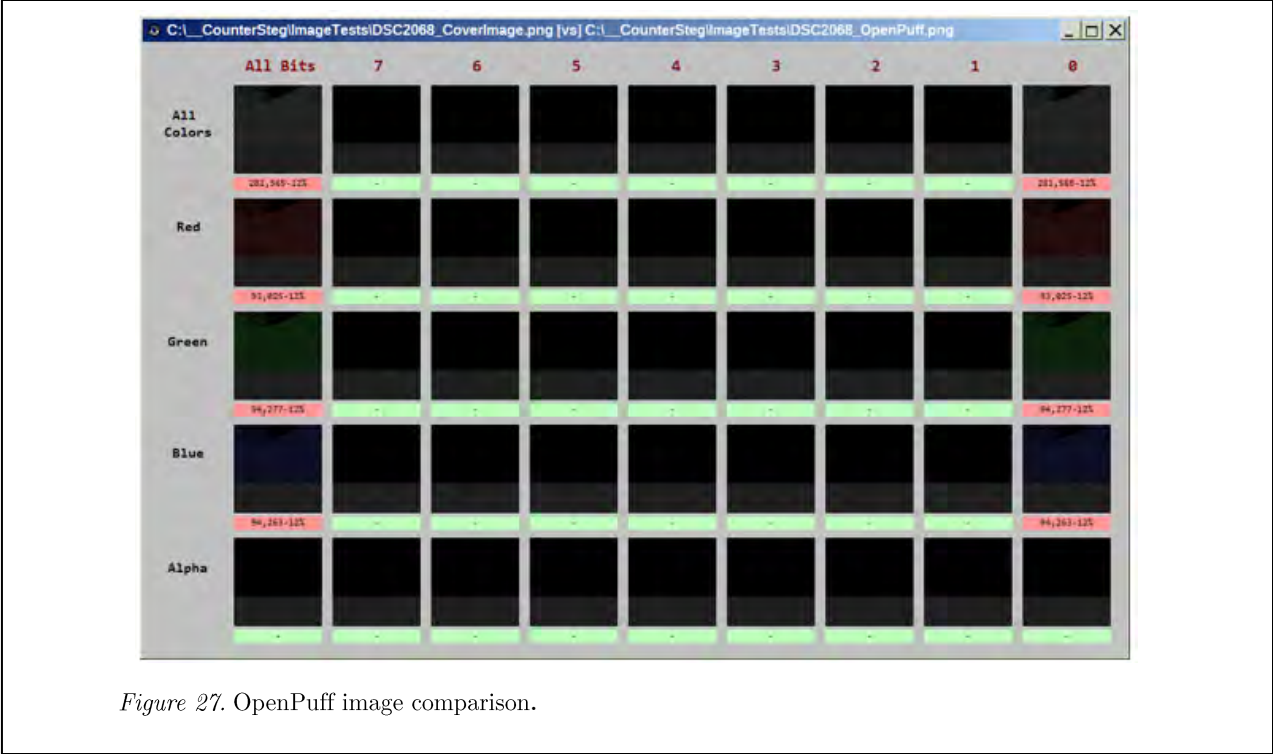


Figure 26. OpenPuff image analysis.

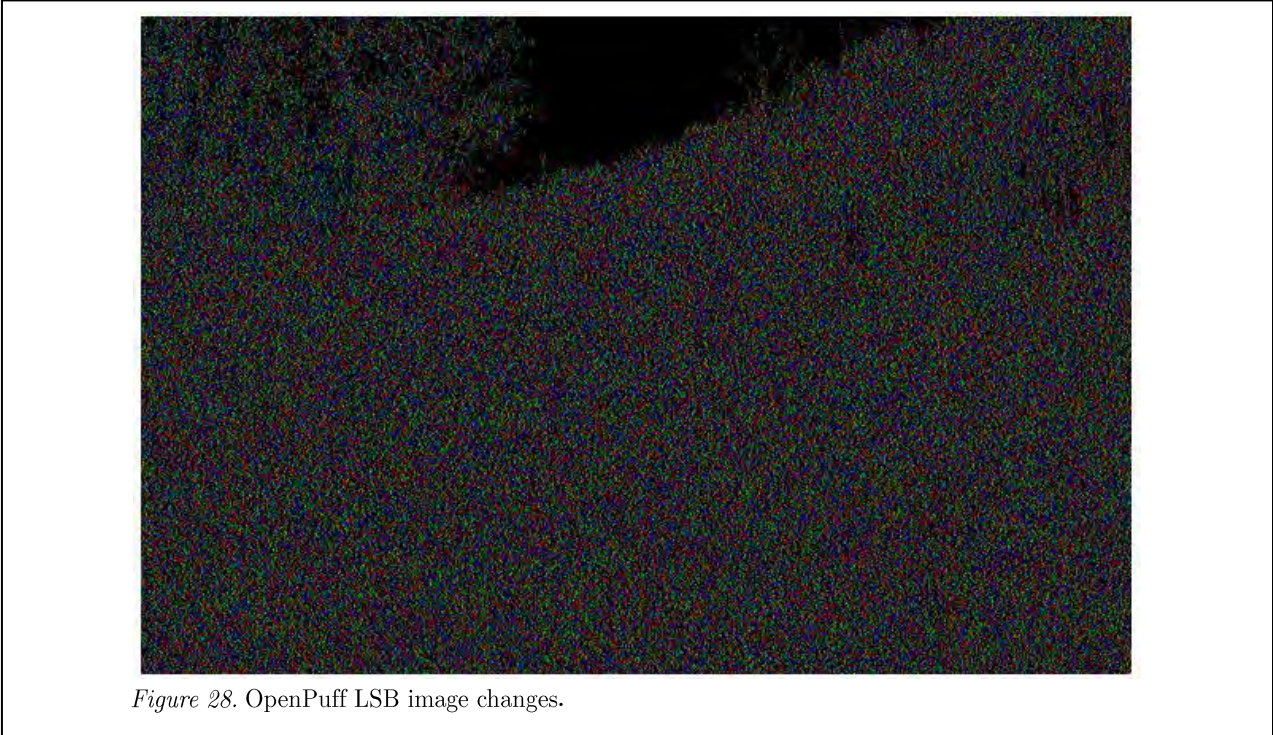
As shown below in Figure 27, OpenPuff only makes changes to the LSB values in the image but does these to far too great an extent at 12%. This leaves the payload image vulnerable to statistical analysis techniques. With a

comparison image in hand, since only LSB values are modified, it is highly probable to conclude steganographic payload has been embedded into the image.



Further, as shown below in Figure 28, the software takes care to avoid saturated color pixel areas. This is commendable; however, without exception, all other areas of the image

are highly modified, possibly with random data. This apparently random data will be more apparent to sophisticated image forensic analysis techniques.



6. THE GOOD

The good steganographic software available in general attempts to do several things. It creates a low payload profile, in our examples below 1% of the pixel values only of the image are modified. This compares to the 12% to 83% of the pixels modified by the other software analyzed previously. Less modifications will correspond to and equal less detectability. Also, it is important to disperse the changes into various areas of the image, specifically areas of higher noise and color variation. Areas of solid colors or pixel saturation should be actively and strongly avoided to lower the possibility and probability of forensic detectability.

6.1 OpenStego

<https://www.openstego.com/index.html>

OpenStego takes care to compress the data before embedding, reducing the overall payload size to about 1% of the image pixels. Also, it disperses the data encoding seemingly randomly throughout the image.

The overall image analysis shown below in Figure 29 provides little for further examination if only the payload image exists. It is likely any known statistical analysis technique will fall flat and fail when trying to determine if any data modifications have been performed on the existing image in hand.

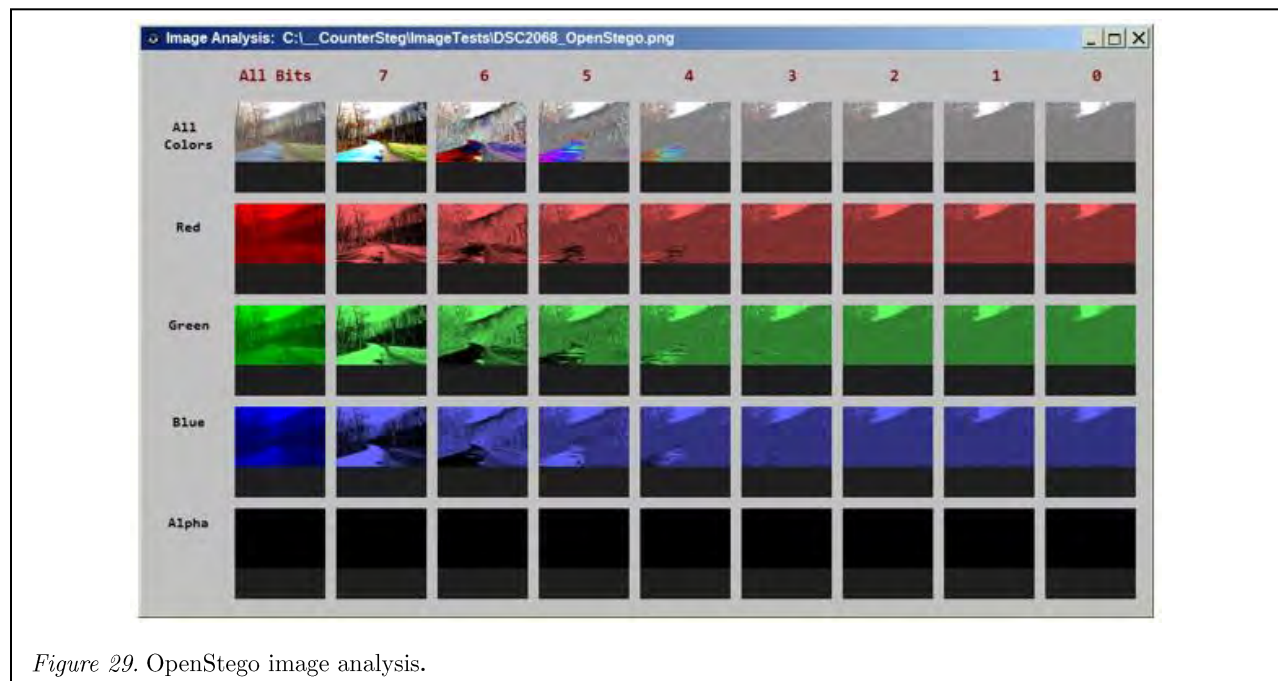


Figure 29. OpenStego image analysis.

The comparison analysis window shown below in Figure 30 indicates only LSB values have been modified, in all three-color channels. The color channels share equally with modifications, at about 1% each. OpenStego,

therefore, is not taking into account relative individual color variations when deciding where to embed data values in various color channels. No modifications are made to the alpha (transparency) color channel.



Below in Figure 31 are shown the LSB modifications in each color channel. Apparently, this is a random distribution not taking into account color variations throughout the image

as previously mentioned. Also, the software does not apparently take into consideration noise levels or saturation levels as well when encoding data.



6.2 6.2 OTP-Steg

<http://www.mausolarsoftware.com/OTP-Steg/>

OTP-Steg receives its name from one-time pad encryption, which is used for encrypting the data in this software before compression and encoding of the data. *OTP-Steg* uses the zlib library to compress all data heavily before encoding. Further, the encoding process looks for and avoids saturated color or solid color

areas of the image. It conducts a noise and variation color analysis of the entire image, to prioritize encoding of LSB data into less statistically detectable areas.

Overall payload image analysis, shown below in Figure 32, shows no features different from a standard photograph in all bit planes. Also, the payload image will be visually indistinguishable from the cover image, as only LSB values are modified.

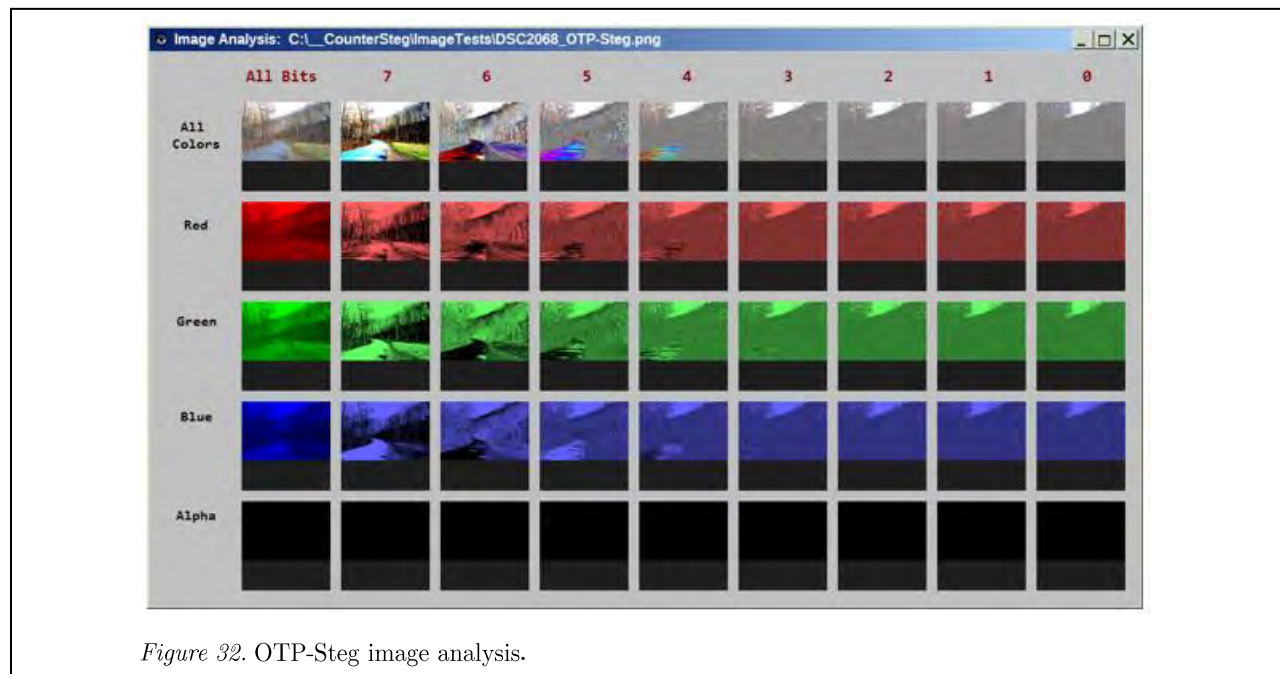


Figure 32. OTP-Steg image analysis.

The comparison to the cover image window shown below in Figure 33 indicates only 1% of the LSB values have been modified. Notice this also in fact varies significantly between color channel, with the red color channel carrying more than double the respective payloads of the

green and blue channels. This is because red color variation varies much more significantly through the image than the green and blue color variations. Thus, data carried in the red channel will be much harder to detect statistically.

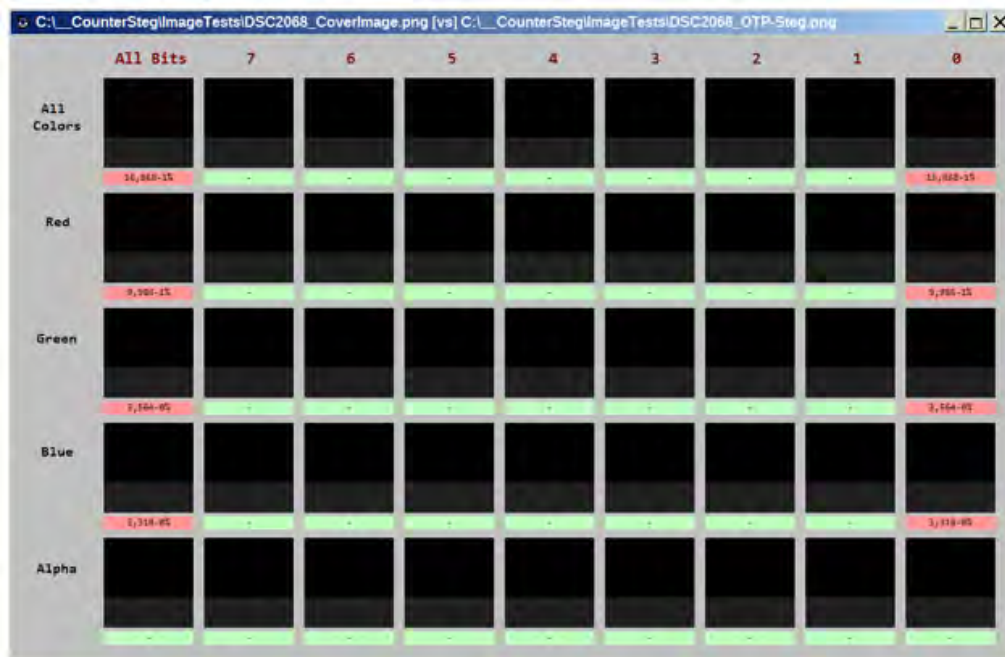


Figure 33. OTP-Steg image comparison.

Shown below in Figure 34 is the image of the specific LSB values changed, in the three visible colors. Changes are dispersed randomly throughout the image, with respect to noise variation. The saturated sky area is completely avoided. Noticeably, red is much more heavily

encoded into the blue and green. This image has a much stronger red component than the other two colors. However, where blue or green are dominant, that is the color encoded into them that respective area. Only one-color channel bit per pixel is allowed to be modified.

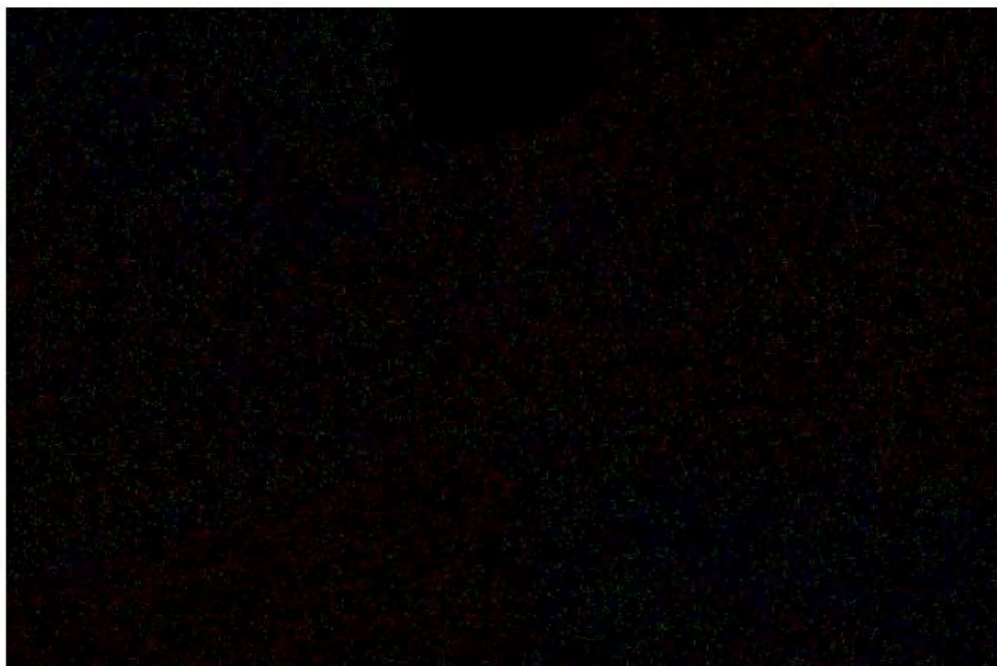


Figure 34. OTP-Steg LSB image changes.

7. SIMILAR IMAGE SEARCH

To aid in locating images for further investigation, the *CounterSteg* software includes facilities for similar image searching on all

machine drives and directories. The similar image search will identify images of identical pixel dimensions and most significant bit values. The value for search of most significant bit similarity percentage is software selectable. The search window design is shown below as Figure 35.

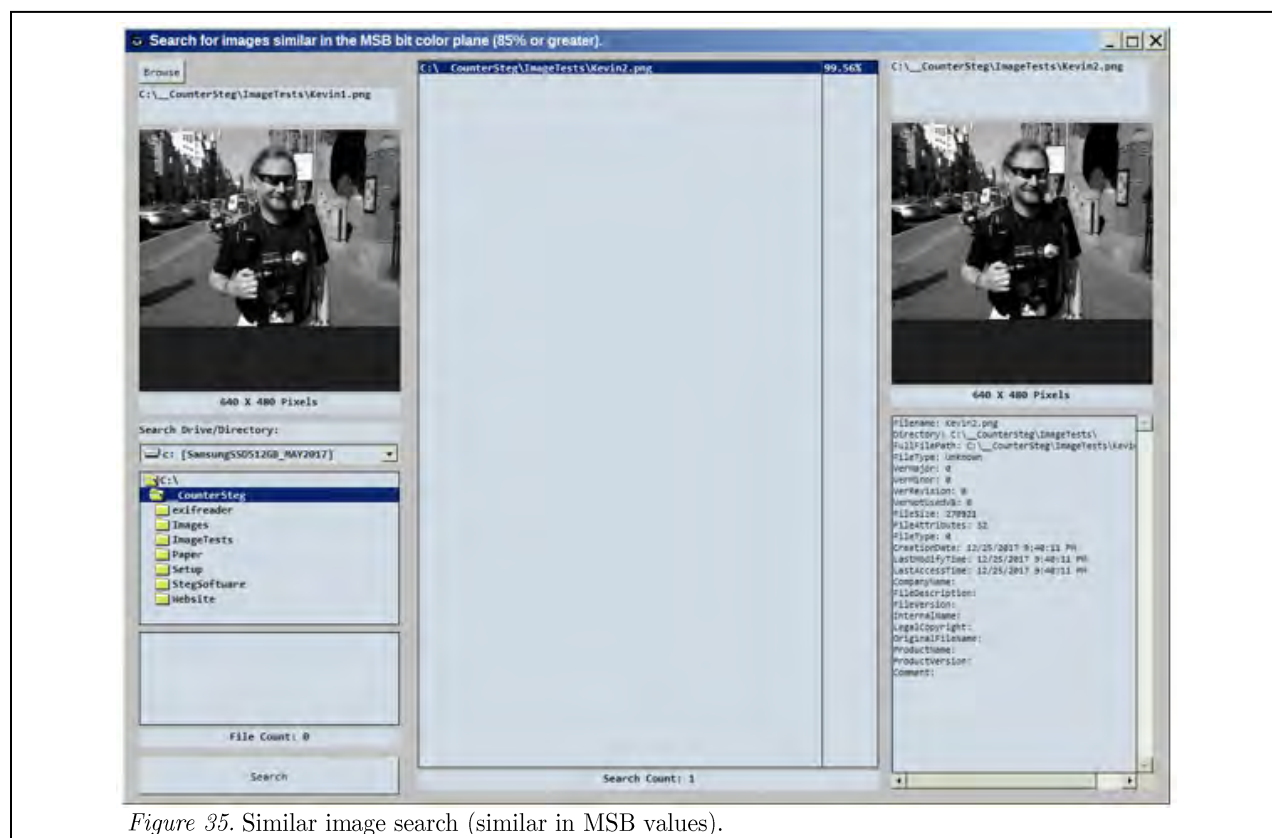


Figure 35. Similar image search (similar in MSB values).

8. FUTURE WORK

We hope to automate the positive identification using image LSB comparison in the future by developing algorithms to conduct analyses. Most likely this will be based on a statistical measure of similarity between pixel color bits above the LSB plane, and difference measures in the LSB plane. In other words, primarily the LSB bits are altered between the image files, preserving the visual appearance, but only altering the data carrying LSB values.

Having such an automated algorithm in hand will greatly assist with the current phenomenon of malware making use of steganography to exfiltrate user and corporate data, such as credit cards, through detection systems designed to thwart such illicit transmissions. The detection system we envision would be based upon "caching" previously known images and comparing transmitted images to ones found in the cache database. Further, web image searches could be conducted by the system to check against images currently in transit. Detection of payload carrying images

could create a red flag to alert security staff to the fact that data exfiltration may be taking place, or that a malware infestation is receiving command and control messages through payload carrying images.

Also, it may be possible to automate part of the process for similar image searching and retrieval using the Google custom search API facilities.

9. CONCLUSION

Positively detecting the use of steganography in digital image files generally results in an unreliable and inconclusive effort. At least by attempting to actively recover the original image, before pixel bit alteration, is estimated to make this procedure much more reliable for positive identification by providing investigative software to allow such a comparison efficiently.

Instead of performing statistical analysis to overall produce dubious results, if steganography is suspected, we posit the investigator would be more effective in simply looking for the original image file for comparison. With comparison software in place, the investigator can assign virtually conclusive attribution. This can be immediately obtained as shown in the numerous examples previously presented in this paper, and useful as evidence in legal proceedings or requests for initial or additional warrants.

REFERENCES

- R. J. Anderson, and F. A. P. Petitcolas, "On the limits of steganography", IEEE Journal of Selected Areas in Communications, vol.16, no.4, pp.474-481, 1998.
- M. Bashardoust, G. B. Sulong, and P. Gerami, "Enhanced LSB image steganography method by using knight tour algorithm, vigenere encryption and LZW compression", International Journal of Computer Science Issues, vol.10, no.2, pp.221-227, 2013.
- A. Bhattacharya, I. Banerjee, and G. Sanyal, "A survey of steganography and steganalysis techniques in image, text, audio and video cover carrier", Journal of Global Research in Computer Science, vol.2, no.4, pp.1-16, 2011.
- Bureau, Pierre-Marc, and Christian Dietrich. "Hiding in Plain Sight." (2015).
- Cabaj, K., et al. "The New Threats of Information Hiding: The Road Ahead." arXiv preprint arXiv:1801.00694 (2018).
- C. K. Chan, and L. M. Chang, "Hiding data in images by simple LSB substitution", Pattern Recognition, vol.37, pp.469-474, 2004.
- A. Cheddad, J. Condell, K. Curran, and P.M. Kevitt, "Digital image steganography: survey and analysis of current methods", Signal Processing, vol. 90, pp.727-752, 2010.
- Chen, Ping, et al. "Advanced or not? A comparative study of the use of anti-debugging and anti-VM techniques in generic and targeted malware." IFIP International Information Security and Privacy Conference. Springer International Publishing, 2016.
- Daryabar, Farid, Ali Dehghantanha, and Hoorang Ghasem Broujerdi. "Investigation of malware defence and detection techniques." International Journal of Digital Information and Wireless Communications (IJDIWC) 1.3 (2011): 645-650.
- S. M. Douiri, M. B. O. Medeni, S. Elbernoussi, and E. M. Souidi, "A new steganographic method for gray scale image using graph coloring problem", Applied Mathematics & Information Sciences, vol.7, no.2, pp.521-527, 2013.
- A. Gangwar, and V. Srivastava, "Improved RGB-LSB steganography using secret key", International Journal of Computer Trends and Technology, vol.4, no.2, pp.85-89, 2013.
- R. S. Gutta, Y. D. Chincholkar, and P. U. Lahane, "Steganography for two and three LSBs using extended substitution algorithm", ICTAT Journal on Communication Technology, vol.4, no.1, pp.685-690, 2013.
- A. Gutub, M. Ankeer, M. Abu-Ghalioun, A. Shaheen, and A. Alvi, "Pixel indicator high capacity technique for RGB image based steganography", in Proceedings of Fifth IEEE International Workshop on Signal Processing and its Applications, 2008, University of Sharjah, U.A.E.
- Gutub, Adnan, et al. "Pixel indicator high capacity technique for RGB image based Steganography." (2008).
- N. Hamid, A. Yahya, R.B. Ahmad, D. Nejim, and L. Kannon, "Steganography in image files: a survey", Australian Journal of Basic and Applied Sciences, vol.7, no.1, pp.35-55, 2013.

- J. He, S. Tang, and T. Wu, "An adaptive steganography based on depth-varying embedding", in Proceedings of 2008 Congress on Image and Signal Processing, 2008, pp.660-663.
- M. Hussain, and M. Hussain, "A survey of image steganography techniques", International Journal of Advanced Science and Technology, vol. 54, pp.113-123, 2013.
- Y. K. Jain, and R. R. Ahirwal, "A novel image steganography method with adaptive number of least significant bits modification based on private stego-keys", International Journal of Computer Science and Security, vol.4, no.1, pp.40-49, 2010.
- M. Juneja, and P.S. Sandhu, "Designing of robust image steganography technique based on LSB insertion and encryption", in Proceedings of International Conference on Advances in Recent Technologies in Communication and Computing, 2009, pp.302-305.
- Kamaldeep, "Image steganography techniques in spatial domain, their parameters and analytical techniques: a review article", IJAIR, vol.2, no.5, pp.85-92, 2013.
- H. B. Kekre, A. A. Athawale, and P. N. Halarnkar, "Increased capacity of information hiding in LSB's method for text in image", International Journal of Electrical, Computer and System Engineering, vol.2, no.4, pp.246-249, 2008.
- Y. K. Lee, G. Bell, S.Y. Huang, R.Z. Wang, and S.J. Shyu, "An advanced least-significant-bit embedding scheme for steganographic encoding", LNCS, vol.5414, 2009, pp.349-360.
- B. Li, J. He, J. Huang, and Y.Q. Shi, "A survey on image steganography and steganalysis", Journal of Information Hiding and Multimedia Signal processing, vol.2, no.2, pp.142-172, 2011.
- D. C. Lou, and C. H. Hu, "LSB steganographic method based on reversible histogram transformation function for resisting statistical steganalysis", Information Sciences, vol.188, pp.346-358, 2012. Application of a large key cipher in image steganography by exploring the darkest and brightest pixels", International Journal of Computer Science and Communication, vol. 3, no.1, pp.49-53, 2012.
- A. R. S. Marcal, and P.R. Pereira, "A steganographic method for digital images robust to RS steganalysis", LNCS, vol.3656, 2005, pp.1192-1199.
- A. Martin, G. Sapiro, and G. Seroussi, "Is image steganography natural", IEEE Transactions on Image Processing, vol.14, no.12, pp.2040-2050, 2005.
- M. K. Meena, S. Kumar, and N. Gupta, "Image steganography tool using adaptive encoding approach to maximize image hiding capacity", International Journal of Soft Computing and Engineering, vol.1, no.2, pp.7-11, 2011.
- A. Mishra, A. Gupta, and D. K. Vishwakarma, "Proposal of a new steganography approach", in Proceedings of International Conference on Advances in Computing, Control, and Telecommunication Technologies, 2009, pp.175-178.
- H. Mathkour, G. M. R. Assassa, A. A. Muharib, and I. Kiady, "A novel approach for hiding messages in images", in Proceedings of International Conference on Signal Acquisition and Processing, 2009, pp.89-93.
- Melanson, Michael. Challenges to Law Enforcement in Detection of Steganography. Diss. Utica College, 2017.
- H. Motameni, M. Norouzi, and A. Hatami, "Labeling method in steganography", World Academy of Science, Engineering and

- Technology, vol. 24, pp.349-354, 2007. , vol. 270, part II, 2012, pp.479-488.
- M. T. Parvez, and A. A. Gutub, "RGB intensity based variable-bits image steganography", in Proceedings of IEEE Asia-Pacific Services Computing Conference, 2008, pp.1322-1327.
- Gandharba Swain et al. / International Journal of Computer Science & Engineering Technology (IJCSET)
- Pevný, Tomáš, et al. "Malicons: Detecting Payload in Favicons." *Electronic Imaging* 2016.8 (2016): 1-9.
- A. P. S. Pharwaha, "Secure data communication using moderate bit substitution for data hiding with three layer security", *IE(I) Journal-ET*, vol.91, pp.45-50, 2010., *International Journal of Security and Its Applications*, vol.6, no.2, pp.1-12, 2012.
- Priyanka, Rama, and P. K. Sahoo. "Scanning Tool for Identification of Image with Malware." *IJACTA* 4.1 (2016): 170-175.
- G. Swain, and S. K. Lenka, "LSB array based image steganography technique by exploring the four least significant bits", *CCIS*
- G. Swain, D. R. Kumar, A. Pradhan, and S. K. Lenka, "A technique for secure communication using message dependent steganography", *International Journal of Computer and Communication Technology*, vol.2, no. 2- 4, pp.177-181, 2010.
- G. Swain, and S. K. Lenka, "Steganography using the twelve square substitution cipher and an index variable", in Proceedings of ICECT, 2011, vol.3, pp.84-88.
- G. Swain, and S. K. Lenka, "A robust image steganography technique using dynamic embedding with two least significant bits", *Advanced Materials Research*, vols. 403-408, pp.835-841, 2012.
- G. Swain, and S. K. Lenka, "A dynamic approach to image steganography using the three least significant bits and extended hill cipher", *Advanced Materials Research*, vols. 403-408 pp.842-849, 2012.
- G. Swain, and S. K. Lenka, "A technique for secret communication by using a new block cipher with dynamic steganography"
- G. Swain, and S. K. Lenka, "A hybrid approach to steganography- embedding at darkest and brightest pixels", in Proceedings of International Conference on Communication and Computational Intelligence, 2010, pp.529-534.
- Walia, Ekta, Payal Jain, and Navdeep. "An analysis of LSB & DCT based steganography." *Global Journal of Computer Science and Technology* (2010).
- Wyke, James. "Breaking the bank (er): Automated configuration data extraction for banking malware." (2015).
- M. A. B. Younes, and A. Jantan, "A new steganography approach for image encryption exchange by using least significant bit insertion", *International Journal of Computer Science and Network Security*, vol.8, no.6, pp.247-254, 2008.
- H. J. Zhang, and H. J. Tang, "A novel image steganography algorithm against statistical analysis", in Proceedings of Sixth International Conference on Machine Learning and Cybernetics, 2007, pp.3884-3888.

